

# On Optimizing Hyperparameters for Quantum Neural Networks

Sabrina Herbst

Vincenzo De Maio

Ivona Brandic\*

## Abstract

The increasing capabilities of Machine Learning (ML) models go hand in hand with an immense amount of data and computational power required for training. Therefore, training is usually outsourced into HPC facilities, where we have started to experience limits in scaling conventional HPC hardware, as theorized by Moore’s law. Despite heavy parallelization and optimization efforts, current state-of-the-art ML models require weeks for training, which is associated with an enormous  $CO_2$  footprint. Quantum Computing, and specifically Quantum Machine Learning (QML), can offer significant theoretical speed-ups and enhanced expressive power. However, training QML models requires tuning various hyperparameters, which is a nontrivial task and sub-optimal choices can highly affect the trainability and performance of the models. In this study, we identify the most impactful hyperparameters and collect data about the performance of QML models. We compare different configurations and provide researchers with performance data and concrete suggestions for hyperparameter selection.

## 1 Introduction

With the exponential increase of data of recent years, Machine Learning (ML) has become a key technology for analytics and pattern recognition [10]. Beyond that, ML models are becoming prevalent in everyday life, as, for example, Large Language Models (LLMs) assist in day-to-day tasks. However, these models require a lot of training time and are associated with a significant  $CO_2$  footprint. Meta’s Llama 2 LLMs require between 184,320 and 1,720,320 GPU hours for pretraining, which is associated with between 31.22 and 291.42  $tCO_2$  equivalents [51] (as a comparison, the EU27 per-capita fossil fuel emissions for 2021 were 6.25  $tCO_2$

equivalents [7]). These factors, together with the recently reached limit of the Von Neumann architecture, motivate the need to scale computational and storage resources to meet the increasing demand of modern ML applications. The most promising path is the integration of Non-Von Neumann architectures, such as Quantum Computing, into the HPC continuum [47].

Quantum computers exploit principles of quantum mechanics, resulting in speed-ups over classical computers for certain computations, and pave the way to entirely novel solutions [14]. Recent quantum computing advances, and the accessibility of quantum machines in the cloud, have opened the doors to quantum computing for researchers across various disciplines [9]. Quantum Machine Learning (QML), i.e., extending the learning paradigm to quantum computers, has emerged as a promising area of research, which could offer significant advantages in terms of runtime, performance, and space efficiency [38].

However, several problems limit the adoption of QML. First, the small number of qubits that is available today limits the input data. Moreover, only Noisy Intermediate-Scale Quantum (NISQ) technology will be available in the near future [41], meaning current and near-term quantum hardware exhibits significant levels of noise, resulting in errors during computation [44]. Furthermore, encoding classical data into a quantum state can be computationally intensive and could cancel out potential runtime benefits [34]. Finally, the model architecture can strongly affect its performance and determining the optimal choice poses an optimization problem with an intractable search space [45]. To this date, there exist only few studies, with limited scope, investigating these hyperparameters.

**Our contribution.** We collect data about QML models by investigating the influence of different hyperparameters on their performance. We select four classical classification datasets, identify the main hyperparameters, and evaluate their impact on the runtime and predictive performance of the models. Further, we study the impact of hardware noise

\*Vienna University of Technology, Vienna, Austria, sabrina.herbst@tuwien.ac.at, {vincenzo,ivona}@ec.tuwien.ac.at

on the different hyperparameters.

**We focus on Quantum Neural Networks (QNNs)**, due to their relevance for near-term hardware [4], and on tabular data as the number of available quantum bits (qubits) limits the input data. We employ the IBM Qiskit package and the included simulators [42] for our evaluation. We choose Qiskit as a programming language, as it is a standard for publicly available state-of-the-art quantum machines, which can be used with quantum backends besides IBM, such as Amazon<sup>1</sup> or AQT<sup>2</sup>. One caveat of using the package is that it allows only limited configuration options, reducing the possibility of adapting the models. Nonetheless, given the limited accessibility of quantum computers, this work will support scientists and practitioners in furthering research in QML with these very machines by providing data and guidelines for hyperparameter tuning. To the best of our knowledge, this work constitutes the biggest publicly available study for QNN architectures to this date.

Our results show that the **optimizer and initialization method constitute the most important hyperparameters for QNNs**. Entangling the feature map can be favorable, but requires efficient initialization to overcome issues during training. The exact entangling strategy has a negligible effect on the performance of the model in our setting. All code and results are available publicly<sup>3</sup>.

We structure the paper as follows: first, we provide preliminary concepts and related work in Section 2. In Section 3 we describe our methodology. The experimental setup and evaluation are discussed respectively in Section 4 and Section 5. Our experimental results are discussed in Section 6, including threats to validity of this work. Finally, we conclude our paper in Section 7.

## 2 Background

### 2.1 Quantum Information Theory

Classical computers work with bits, which are either in state 0 or 1. In quantum computing, the so-called *computational basis* is composed of the states  $|0\rangle$  and  $|1\rangle$ . A single-qubit quantum state  $|\phi\rangle$  forms a linear combination

of these two states, called a *superposition*, represented as  $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ , with  $\alpha, \beta \in \mathbb{C}$  and  $|\alpha|^2 + |\beta|^2 = 1$ .  $|\phi\rangle$  is in both states at the same time, but takes the value of  $|0\rangle$  with probability  $|\alpha|^2$  and  $|1\rangle$  with probability  $|\beta|^2$ , once we measure the state. *Quantum parallelism* harnesses this phenomenon, allowing quantum computers to concurrently represent multiple states, rather than needing to evaluate each state separately.

Another peculiarity of quantum computing is *quantum entanglement*, where multiple qubits are correlated, such that performing an action on one qubit will impact all qubits it is entangled with as well. Superposition and entanglement are usually exploited in quantum algorithms to achieve significant speed-ups.

### 2.2 Quantum Machine Learning

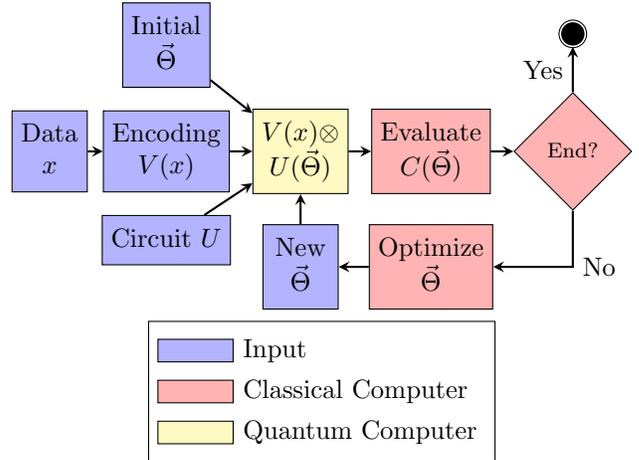


Figure 1: Quantum Neural Networks

Working with classical data on quantum computers requires an effective strategy for encoding the information into a quantum state [38, p.25-28]. Exploiting the unique characteristics of a quantum computer increases the chance of obtaining an advantage, therefore, it is recommended to choose an encoding that is hard to obtain or simulate on a classical computer [12].

Similarly to classical ML, several classification algorithms exist, including Quantum Support Vector Machines [43] or K-Means Clustering [31], both with an exponential speed-up compared to the classical algorithms. However, Quan-

<sup>1</sup><https://github.com/qiskit-community/qiskit-braket-provider>. Accessed: 04.01.2024

<sup>2</sup><https://github.com/qiskit-community/qiskit-aqt-provider>. Accessed: 04.01.2024

<sup>3</sup>[https://github.com/sabrinaherbst/hyperparameters\\_qnn](https://github.com/sabrinaherbst/hyperparameters_qnn)

tum Neural Networks (QNNs) have come up as one of the leading algorithms that could achieve a quantum advantage on NISQ technology, due to their simple architecture and training process [4].

QNNs are depicted in Figure 1. Initially, classical data is encoded into a quantum state using a feature encoding method  $V$ . The inputs to the algorithm are a parametrized quantum circuit  $U$  (ansatz), which is a sequence of trainable unitary transformations, and associated initial parameters  $\bar{\Theta}$ . The goal is to find parameters  $\bar{\Theta}$  that minimize the cost function  $C$ , i.e., cross-entropy for classification. After executing the circuit, the parameters  $\bar{\Theta}$  are optimized on a classical computer using an optimizer, such as ADAM [24] or COBYLA [40]. The parameters are then transferred to the quantum computer, where the circuit is executed again. The process is repeated until a certain termination criterion is reached (i.e., maximum number of iterations, tolerance threshold). The different choices for feature map, ansatz and optimizer can highly impact the results.

Currently, one of the biggest challenges in QNNs is mitigating the **Barren Plateau (BP)** phenomenon. BPs cause the gradient of the cost function to vanish exponentially in problem size, rendering optimization of the parameters difficult [4]. In particular, randomly initialized deep circuits are subject to these plateaus [33]. This has been extended for shallow ones and linked to the structure of the cost function [5]. Furthermore, BPs have been linked to expressivity [18], entanglement [37] and noise [53]. Ways to overcome them include different parameter initialization strategies [57, 13, 54], or limiting the size of the accessible Hilbert space through, e.g., problem-specific ansatzes [52, 15].

## 2.3 Related Work

In [48], the influence of the type of data, classical transformations on data, and feature encoding techniques on QML models, focusing on amplitude encoding and the ZZFeatureMap, are investigated. The authors generate synthetic datasets using different transformations and apply various rotations to them. They observe that amplitude encoding is less affected by transformations than the ZZFeatureMap is. As rotating the dataset improves some models, the authors propose a strategy of rotating it to create a different representation. The approach aligns with the idea of [46], suggesting that appropriately scaling the data may be beneficial in QML.

Mancilla and Pere employ Classical and Quantum ML in [32] and compare different feature reduction techniques. They find that Linear Discriminant Analysis (LDA) outperforms Principal Component Analysis (PCA) for QML models. Hanco-Quispe et al. report similar results in [16].

Joshi et al. [21] investigate local optimizers in QNNs using EfficientSU2 as the ansatz and ZFeatureMap as the feature map. They compare the performance using AQGD and COBYLA to classical machine learning models in analyzing sentiment data. The QNN with AQGD outperforms the other models.

Moreover, the authors in [50] compare the performances of a hybrid deep learning model and a QNN. They consider different feature map, ansatz and optimizer combinations for the QNN and find significant differences in performance.

Piatrenka and Rusek [39] perform experiments using different configurations for the iris dataset. They experiment using ZFeatureMap, ZZFeatureMap and a PauliFeatureMap with custom Pauli gates. Regarding the ansatz, they only experiment using the RealAmplitudes one, but vary the repetitions. For the optimizer, they consider SPSA, COBYLA and SLSQP. They find that one repetition of the PauliFeatureMap and two repetitions of the RealAmplitudes ansatz work best. Furthermore, they report that COBYLA and SLSQP outperform SPSA. The authors run their experiments on simulators and a real quantum computer and find big performance differences between the executions.

Moreover, Katyayan and Joshi propose a model for classifying questions in [22]. They use a TwoLocal ansatz and experiment with all three of Qiskit’s feature maps. They find that the PauliFeatureMap works best, that a small depth is advantageous, and that the features strongly influence the output.

Finally, Joshi et al. compare classical ML models to quantum ones on sentiment analysis in [20]. They find that quantum models slightly outperform the classical models in their configuration. They employ a PauliFeatureMap using custom Pauli gates and COBYLA as the optimizer. The EfficientSU2 ansatz with 100 epochs outperforms the other models.

Other works, such as [25, 2] define Automated Machine Learning (AutoML) for QML by proposing algorithms for automatically tuning the QML hyperparameters.

## 3 Methodology

In the following, we describe our methodology, the chosen datasets and selected QNN hyperparameters.

### 3.1 Data

We choose datasets with low dimensionality, due to the limited qubits on the IBM Quantum Open Access Plan<sup>4</sup>.

We consider (1) the KDD Cup 1999 dataset [30], a well-known public classification dataset for intrusion detection, (2) the Cover Type dataset, whose goal is to predict the cover type of forest squares based on cartographic attributes [3], (3) the Glass Identification dataset [11], used to identify types of glass for forensics at crime scenes, and (4) the Rice dataset, for binary classification between two types of rice [26, 6]. All datasets are available on the UCI Machine Learning repository [23].

The datasets offer diverse characteristics. While the Rice dataset is only for binary classification, both Glass Identification and Cover Type have seven target categories and the KDD Cup dataset distinguishes between 22 different attacks. The Glass Identification dataset consists of only 214 samples, whereas the other datasets exceed our maximum training samples of 400. We apply a One-Hot Encoding for categorical features and compare PCA and LDA for reducing the dimensionality to seven features. We do not employ LDA for the Rice dataset, as it would lead to only one feature. As the encoding we choose employs one qubit per feature, entangling the qubits would not be possible anymore.

### 3.2 Optimizers

We employ COBYLA [40], SPSA [49] and Nelder-Mead [36] as optimizers. Both COBYLA and SPSA are extensively used in the QML literature, as mentioned in Section 2.3. We also select Nelder-Mead due to its popularity on different quantum optimization problems [55, 35, 29].

SPSA is a gradient-based method, which is known to work well in the presence of noise<sup>5</sup>. It calculates the objective

<sup>4</sup>Note: The plan was changed after we conducted the experiments. IBM used to offer free seven-qubit-machines for the public, now larger machines are available but with a ten-minute time constraint per month.

<sup>5</sup><https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.SPSA.html>. Accessed 17.06.2023

function using only two measurements, irrespective of the number of parameters involved in the optimization problem. Nelder-Mead is a heuristic gradient-free method which performs unconstrained optimization, and is based on the simplex algorithm. COBYLA is gradient-free as well, but uses trust regions instead and can be used for constrained optimization. By utilizing optimizers with diverse characteristics, we aim to identify which type of optimizer is likely to lead to high accuracy for QML models.

### 3.3 Ansatzes

Ansatzes are characterized by the circuit structure, i.e., the sequence of operations and the employed entanglement.

#### 3.3.1 Circuit Structure

The ansatz defines the trainable parameters of the circuit. We select four popular ansatzes from the Qiskit package, which are widely used in QML [21, 22, 39] and relevant for near-term hardware: PauliTwoDesign, RealAmplitudes, EfficientSU2 and TwoLocal. The ansatzes are built by repeatedly applying alternating rotation and entanglement blocks. The number of repetitions can be tuned and defines the circuit depth.

$R_{X|Y|Z}$  gates are used for rotations around the X, Y and Z axes of the Bloch sphere, which is a geometrical representation of a qubit, the angle defined by the learnable parameters  $\theta$ . Furthermore, two qubits are entangled using entanglement gates, which are either controlled-X gates or controlled-Z gates. They perform an X/NOT operation ( $|0\rangle \rightarrow |1\rangle$ ,  $|1\rangle \rightarrow |0\rangle$ , see Equation 1) or Z ( $|0\rangle \rightarrow |0\rangle$ ,  $|1\rangle \rightarrow -|1\rangle$ , see Equation 2) transformation on one qubit (called the target), should the other one (control) be in state  $|1\rangle$ .

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (1) \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (2)$$

For better illustration, we show the RealAmplitudes ansatz in Figure 2. The  $R_Y$  operations rotate the individual qubits around the y-axis of the Bloch sphere, with an angle defined by the trainable parameter  $\theta$ . The operations in between are the controlled-NOT gates (CNOT), which entangle the qubits. The target qubit is denoted as  $\oplus$  and the control qubit as  $\bullet$ . The rotation and entanglement blocks are repeated three times in this case, making up an ansatz

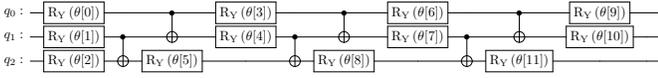


Figure 2: RealAmplitudes Ansatz

of depth three. The goal of training is to find the optimal angles  $\vec{\theta}$ .

We experiment with the entanglement hyperparameter, leaving all others to their default values. Additionally, for TwoLocal, as suggested in the Qiskit documentation, we utilize the  $R_Y$  rotation and controlled-X entanglement block.

We experiment using Qiskit’s default initialization (uniform in range  $[0, 1]$ ), the beta distribution, as proposed in [27], and the normal distribution with  $\sigma^2 = \frac{1}{L}$ , where  $L$  is the number of layers [56].

### 3.3.2 Qubit Entanglement

Another design choice is how to entangle qubits. Possible choices are full (each qubit is entangled with all others), linear (qubit  $q_i$  is entangled with  $q_{i+1}$ , with  $i \in [0, N - 2]$  where  $N$  is the number of qubits), circular (linear, plus  $q_{N-1}$  is entangled with  $q_0$ ), or sca (shifted-circular-alternating).

In sca entanglement, the target and control qubit of the entanglement gates are swapped in each block. Furthermore, the entanglement strategy is similar to circular. Qubits  $q_{N-1}$  and  $q_0$  are entangled in the first iteration before all others. In the next iteration, first  $q_0$  and  $q_1$  are entangled, then  $q_{N-1}$  and  $q_0$ , and then the rest. The entanglement between  $q_{N-1}$  and  $q_0$  is "shifted" in every iteration. We exclude reverse-linear due to its similarity to the linear hyperparameter choice.

TwoLocal additionally supports pairwise entanglement (in even layers, qubit  $q_i$  is entangled with  $q_{i+1}$ , in uneven ones with  $q_{i-1}$ ).

## 3.4 Encoding

Working with classical data on quantum computers requires encoding the information into a quantum state. A very simple example of data encoding is the so-called *basis encoding*. The value, for example the number 5, is transformed into a binary string (101) and each bit is trans-

lated into the quantum counterpart, requiring three qubits:  $5 \rightarrow 101 \rightarrow |1\rangle |0\rangle |1\rangle = |101\rangle$  [38, p.25-26].

We employ a feature map based encoding, as proposed by Havlicek et al. [17]. It uses a quantum feature map to transform the features and aims to exploit the dimensionality of the Hilbert space.

Equation 3 shows the so-called Pauli expansion circuit.  $i$  refers to the imaginary unit and  $n$  to the number of qubits.  $P$  is a set of Pauli gates, consisting of  $X$ ,  $Y$  (see Equation 4),  $Z$ , and identity gates  $\mathbf{I}$ .

$$U_{\Phi(x)} = \exp(i \sum_{S \subseteq [n]} \phi_S(x) \prod_{i \in S} P_i) \quad (3)$$

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad (4)$$

$S \in \binom{[n]}{k}$  describes the connections between the qubits. The connections are influenced by  $k \in \{1, \dots, n\}$ , the entanglement hyperparameter and the Pauli matrices.  $\phi_S(x)$  refers to a non-linear function. Equation 5 shows the default non-linear function in Qiskit.<sup>6</sup>

$$\phi_S(x) = \begin{cases} x_0, k = 1 \\ \prod_{j \in S} (\pi - x_j), \text{ else} \end{cases} \quad (5)$$

The tensor product of  $n$  Hadamard gates (see Equation 6) is then applied to the input qubits (all in state  $|0\rangle$ ), which puts them in an equal superposition of all possible states (if we measure we have the same probability for all  $2^n$  states). Afterward, the Pauli expansion circuit is applied. The two operations can be repeated arbitrary times. Equation 7 shows the final feature map with two repetitions.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (6)$$

$$|\Phi(x)\rangle = U_{\Phi(x)} H^{\otimes n} U_{\Phi(x)} H^{\otimes n} |0\rangle_n \quad (7)$$

Qiskit provides two variants: ZZFeatureMap and ZFeatureMap. The ZFeatureMap sets  $k = 1$  and  $P_0 = Z$ , hence not entangling the qubits, and the ZZFeatureMap sets  $k = 2$ ,  $P_0 = Z$  and  $P_{0,1} = ZZ$ . The implementations use one qubit for every feature present in the data. We leave all hyperparameters of the ZFeatureMap to their default values. We optimize the entanglement strategy for the ZZFeatureMap, which allows the same options as TwoLocal.

<sup>6</sup><https://qiskit.org/documentation/stubs/qiskit.circuit.library.PauliFeatureMap.html>. Accessed 20.06.2023.

## 4 Experimental Setup

### 4.1 Hardware Specifications

Quantum simulators have the advantage of being capable of simulating a perfect quantum computer on a classical one without any noise, allowing to create baselines to compare the results to experiments executed on real quantum machines. By incorporating noise models in simulators, one can obtain a more accurate idea of the results on a real quantum computer [19].

Thus, we conduct our experiments using quantum simulators, specifically, we employ the simulators provided by Qiskit Aer. We first run the experiments on a perfect simulator, before incorporating noise models generated from real machines. We choose a noise model from IBM’s 7-qubit quantum computer called “Perth”<sup>7</sup>.

While quantum simulators have several advantages, e.g., availability and controlled noise, phenomena that are inherently quantum (i.e., entanglement) are computationally intense to simulate on a classical computer, which makes comparing runtimes more difficult. Therefore, we analyze and compare the runtime mainly for aspects which can also be expected on real quantum hardware (e.g., number of parameters or circuit depth).

We run our experiments on a Intel(R) Xeon(R) CPU E5-2623 v4 (16 cores @ 2.60GHz) server with 128GB RAM and a Debian/GNU Linux 11 OS. We use Python v3.11.7 and the IBM Qiskit framework v0.43 [42] for the implementation.

### 4.2 Experiments

We evaluate all hyperparameter configurations of Section 3, summing up to 1512 configurations per dataset and noise configuration. We compare our setup to related work in Table 1.

Concerning optimizers, to assess the convergence behavior and set an appropriate iteration count for every dataset, noise setting and optimizer, we plot the training loss at different iterations. For COBYLA, convergence typically occurs within a range of 250 to 500 iterations. The SPSA optimizer converges within 125 to 300 iterations, and the Nelder-Mead optimizer within 100 to 250 iterations. Additionally, we apply early stopping with a tolerance value of

<sup>7</sup>Available through [https://docs.quantum.ibm.com/api/qiskit/qiskit.providers.fake\\_provider.FakePerth](https://docs.quantum.ibm.com/api/qiskit/qiskit.providers.fake_provider.FakePerth). Accessed 16.01.2024

0.1 for the loss function, which is supported by COBYLA and Nelder-Mead.

Moreover, SPSA and Nelder-Mead work by probing solutions (which is done in so-called steps) and only move on to the next iteration if they accept the step. We define the termination criterion based on the number of iterations, however, in the results section, we analyze the runtime of these two optimizers based on the number of steps they took. To avoid any confusion with the reported numbers, we want to actively highlight that the terms iteration and step are not used interchangeably in our analysis.

Furthermore, we activate the adaptive hyperparameter option to improve convergence of Nelder-Mead, which adapts the algorithm’s parameters to the problem dimensionality, since the optimizer struggles to find a reasonable path in some scenarios.

Since the runtime increases quickly with more data, we limit the samples used in the experiments to a maximum of 400 samples for training and 250 for testing the final models, which limits the runtime to ten hours. The trained models are evaluated using the accuracy and weighted f-1 score.

## 5 Results

To better visualize the results, we define the set of best and worst configurations as those that are within 10% accuracy of the best or worst one overall. Furthermore, we study how noise affects the configurations by running a pairwise analysis, comparing them with and without noise directly.

To ensure statistical significance, we perform non-parametric statistical tests with a significance level of 5%. In particular, we choose a Friedman Test for dependent distributions of more than two groups and the Wilcoxon Signed Rank test for two dependent distributions. Moreover, we use the Kruskal-Wallis test for independent distributions of more than two groups and the Mann-Whitney-U test for independent distributions of two groups.

In the following, we distinguish between *observations*, which are general patterns we identify and *main findings*, which are consistent patterns that constitute key insights into QNN training.

### 5.1 Overview

The accuracy for the noiseless and noisy KDD Cup experiments ranges from 0%-97%. The top five configurations

	#Dataset	Prepr.	Feat. Map	F. Ent.	Ansatz	A. Ent.	Opt.	Init.	Noiseless	Noisy
[48]	5	✓	✓						✓	
[32]	2	✓							✓	
[16]	1	✓							✓	
[21]	1						✓			✓
[50]	1		✓		✓		✓		✓	
[39]	1		✓		✓		✓		✓	✓
[22]	1	✓	✓		✓					✓
[20]	1				✓		✓			✓
<b>This work</b>	<b>4</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>

Table 1: Comparison of Studies Investigating QNN Hyperparameters

Acc.	F-1	Time [s]	Ansatz/Entanglement	Optimizer	Feature Map/Ent.	Prepr.	Init
<b>Noiseless</b>							
0.972	0.965	3848	EfficientSU2/sca	SPSA	ZZFeatureMap/sca	LDA	Beta
0.968	0.961	3740	EfficientSU2/sca	SPSA	ZZFeatureMap/linear	LDA	Beta
0.964	0.960	3413	EfficientSU2/sca	COBYLA	ZZFeatureMap/circular	LDA	Beta
0.964	0.959	3022	EfficientSU2/sca	COBYLA	ZZFeatureMap/pairw.	LDA	Beta
0.972	0.958	2598	RealAmplitudes/circular	SPSA	ZFeatureMap	LDA	Beta
<b>Noisy</b>							
0.968	0.954	7300	TwoLocal/linear	COBYLA	ZFeatureMap	LDA	Beta
0.960	0.950	6462	TwoLocal/sca	COBYLA	ZFeatureMap	LDA	Beta
0.964	0.948	6248	TwoLocal/pairw.	COBYLA	ZFeatureMap	LDA	Beta
0.960	0.944	6947	RealAmplitudes/linear	COBYLA	ZFeatureMap	LDA	Beta
0.960	0.944	8569	EfficientSU2/linear	SPSA	ZFeatureMap	LDA	Beta

Table 2: KDD Cup: Top Five Configurations

according to the f-1 score in the different settings are shown in Table 2. We choose the f-1 score as the final criterion, as it represents both Precision and Recall equally, rather than just the ratio of correctly predicted samples. All use LDA for preprocessing and beta distribution initialization. Most of the best configurations in the noiseless setting use EfficientSU2 and ZZFeatureMap. In the noisy one, TwoLocal and ZFeatureMap are more frequent. The mean difference between the configurations with and without noise is at 10% accuracy, with a standard deviation of 11%. The minimum difference lies at 0%, the maximum at 67%.

Both results for the noisy and noiseless Cover Type experiments range from 2%-61% accuracy. We show the top five configurations from both settings in Table 3. Again, the noisy configurations are slightly worse, all use LDA for dimensionality reduction and most configurations use beta distribution initialization and ZZFeatureMap in both set-

tings.

The mean difference in accuracy between noisy and noiseless configurations is 4% with a standard deviation of 3%. Figure 3 shows that more than 75% of configurations have a difference < 8%, the maximum being at 30%. Noisy simulators lead to significantly better results for the Cover Type dataset. However, the best models in the noiseless setting outperform the best ones in the noisy setting.

The results from the Rice dataset are shown in Table 4. The noiseless results range from 7%-91% accuracy, while the noisy ones range from 9%-90%. The configurations with and without noise differ 3% on average in accuracy, with a standard deviation of 3%. We again find configurations with exactly the same performance, however, the maximum difference lies at 36%. Only ZFeatureMap configurations with beta distribution initialization are among the best ones in both settings.

Acc.	F-1	Time [s]	Ansatz/Entanglement	Optimizer	Feature Map/Ent.	Prepr.	Init
<b>Noiseless</b>							
0.612	0.612	2066	EfficientSU2/circular	COBYLA	ZZFeatureMap/linear	LDA	Beta
0.608	0.612	2097	EfficientSU2/linear	SPSA	ZZFeatureMap/sca	LDA	Beta
0.612	0.610	886	PauliTwoDesign/full	COBYLA	ZZFeatureMap/pairw.	LDA	Beta
0.616	0.609	1422	RealAmpl./linear	SPSA	ZFeatureMap	LDA	Nor.
0.612	0.605	2100	EfficientSU2/circular	SPSA	ZZFeatureMap/circular	LDA	Beta
<b>Noisy</b>							
0.608	0.600	1867	TwoLocal/linear	SPSA	ZZFeatureMap/pairw.	LDA	Beta
0.616	0.593	2031	RealAmpl./linear	COBYLA	ZFeatureMap	LDA	Beta
0.588	0.592	2493	RealAmpl./circular	COBYLA	ZZFeatureMap/circular	LDA	Beta
0.596	0.592	3190	PauliTwoDesign/full	COBYLA	ZZFeatureMap/linear	LDA	Beta
0.596	0.588	1922	TwoLocal/full	COBYLA	ZZFeatureMap/linear	LDA	Beta

Table 3: Cover Type: Top Five Configurations

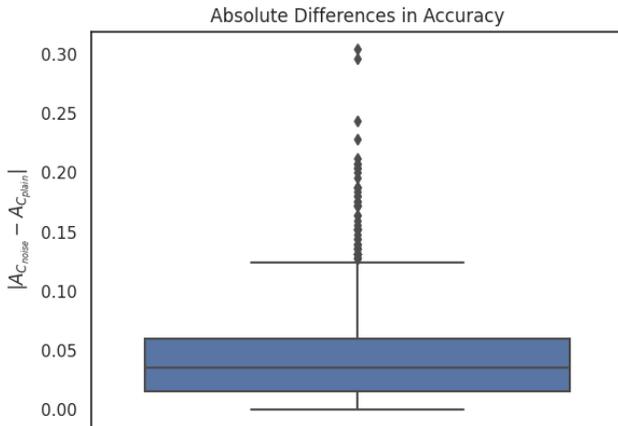


Figure 3: Cover Type: Absolute Difference in Accuracy

We show the top five configurations with and without noise for the Glass Identification dataset in Table 5. Our models had problems fitting the data, with the highest accuracy at only 62%, which could be due to the limited samples. The results without noise range from 2%-62%, and with noise from 0%-53% accuracy. The mean accuracy difference between configurations with and without noise is 7%, with a standard deviation of 6%. While the minimum difference is at 0%, the maximum is at 34%.

## 5.2 Initialization

**Main Finding 1:** Beta initialization performs significantly better than both normal and uniform initialization in all settings. We find no significant differences between normal and uniform initialization in most settings.

Therefore, the lower bounds on the magnitude of the gradient when using the normal distribution for initialization from [56], did not lead to significantly better results in our experiments. We give a detailed overview of the significance tests in the appendix.

Figure 4 shows the performance of the different initialization methods for the Cover Type dataset. Our results are therefore consistent with the conjecture from [27], that initializing using the beta distribution can lead to better trainability. Our results extend their work by applying large-scale real-world datasets beyond the Wine and Iris datasets tested in their study.

Nonetheless, we want to point out that, for the Cover Type dataset, in the noiseless and noisy setting, 8% and 5% of the beta initialized configurations, compared to 1% and < 1% of uniform and normal configurations, are in the set of worst configurations. Therefore, while leading to significantly better results for some configurations, beta initialization can impact the results negatively as well. However, we find this pattern to be dependent on the dataset, as the pattern for the other datasets is not as clear.

When comparing the noiseless and noisy settings, we find

Acc.	F-1	Time [s]	Ansatz/Entanglement	Optimizer	Feature Map/Ent.	Prepr.	Init
<b>Noiseless</b>							
0.916	0.915	612	TwoLocal/pairw.	COBYLA	ZFeatureMap	PCA	Beta
0.904	0.902	1411	RealAmplitudes/linear	SPSA	ZFeatureMap	PCA	Beta
0.900	0.899	668	RealAmplitudes/circular	COBYLA	ZFeatureMap	PCA	Beta
0.896	0.895	688	RealAmplitudes/sca	COBYLA	ZFeatureMap	PCA	Beta
0.896	0.895	8023	TwoLocal/sca	COBYLA	ZFeatureMap	PCA	Beta
<b>Noisy</b>							
0.908	0.907	7470	PauliTwoDesign	SPSA	ZFeatureMap	PCA	Beta
0.896	0.895	4785	RealAmplitudes/sca	COBYLA	ZFeatureMap	PCA	Beta
0.896	0.895	4518	TwoLocal/pairw.	COBYLA	ZFeatureMap	PCA	Beta
0.896	0.894	4573	TwoLocal/sca	COBYLA	ZFeatureMap	PCA	Beta
0.892	0.890	9214	EfficientSU2/full	COBYLA	ZFeatureMap	PCA	Beta

Table 4: Rice: Top Five Configurations

Acc.	F-1	Time [s]	Ansatz/Entanglement	Optimizer	Feature Map/Ent.	Prepr.	Init
<b>Noiseless</b>							
0.604	0.571	608	EfficientSU2/linear	COBYLA	ZZFeatureMap/sca	PCA	Beta
0.627	0.561	470	EfficientSU2/sca	COBYLA	ZFeatureMap	LDA	Beta
0.558	0.546	641	TwoLocal/linear	SPSA	ZZFeatureMap/circular	PCA	Beta
0.581	0.538	302	EfficientSU2/circular	COBYLA	ZFeatureMap	LDA	Beta
0.534	0.535	663	RealAmplitudes/sca	SPSA	ZZFeatureMap/circular	PCA	Beta
<b>Noisy</b>							
0.534	0.519	342	RealAmplitudes/sca	COBYLA	ZZFeatureMap/circular	LDA	Beta
0.488	0.497	3590	EfficientSU2/full	COBYLA	ZZFeatureMap/full	PCA	Beta
0.511	0.487	1959	TwoLocal/sca	SPSA	ZZFeatureMap/linear	PCA	Beta
0.511	0.471	1853	PauliTwoDesign	COBYLA	ZFeatureMap	PCA	Nor.
0.488	0.465	503	TwoLocal/pairw.	SPSA	ZFeatureMap	LDA	Nor.

Table 5: Glass Identification: Top Five Configurations

that all initialization methods work significantly better without noise for KDD Cup. Beta initialization works better in the noiseless setting for the Rice dataset, while we find no difference for the other initialization methods and noise settings. Only beta initialization works significantly better without noise for Cover Type and Glass Identification, while uniform leads to better results when noise is introduced. Normal initialization for Glass Identification also works significantly better with noise.

Furthermore, uniform initialization takes significantly less time in the Cover Type noiseless settings, and in all KDD Cup and Glass Identification settings, than beta and normal do. For KDD Cup and Cover Type, we find that this is mainly due to it stopping between 100-450 steps earlier for

Nelder-Mead configurations. For Glass Identification, we were not able to pin it down to one factor.

Normal initialization is significantly faster than beta in all settings except for the Rice dataset, which is explained by COBYLA requiring between 10 and 50 more iterations for beta than for normal. We find no significant runtime differences for the noiseless Rice dataset, but uniform initialization takes significantly longer than both others in the noisy setting. We find no clear patterns explaining the behavior, however.

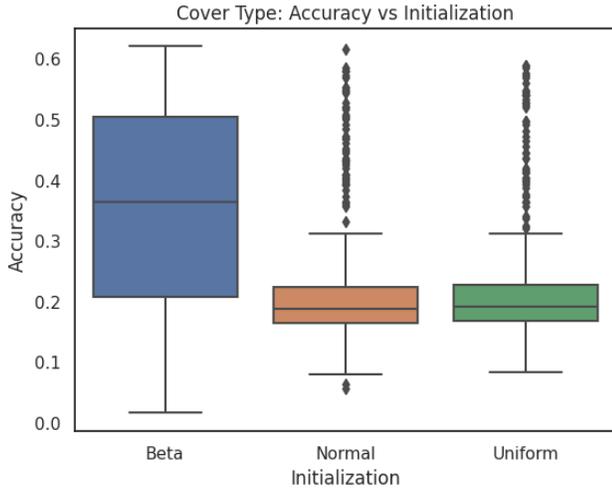


Figure 4: Cover Type, Noiseless: Initialization vs. Accuracy

### 5.3 Optimizer

**Main Finding 2:** SPSA and COBYLA significantly outperform Nelder-Mead in all settings. We find no consistent performance patterns for COBYLA and SPSA, but COBYLA takes significantly less time in all settings.

Nelder-Mead fails to provide a reasonable optimization path, which can be seen from convergence plots such as the one shown in Figure 5. The only time we find Nelder-Mead in the set of best configurations is in the Glass Identification noisy setting.

While the algorithm is commonly used, there are relatively few theoretical studies on convergence [1]. Several inefficiencies and pitfalls are known, however, the authors in [28] identify several reasons why it is still commonly used. First, it is a fairly simple algorithm and, secondly, improves very quickly over current solutions even though it may not converge to optima. Finally, if the method works, it requires few function evaluations, which is particularly relevant for compute-expensive cost functions, where the derivatives are difficult or even impossible to compute.

COBYLA is significantly better than SPSA for the KDD Cup noiseless setting, whereas SPSA significantly outperforms COBYLA on all Cover Type and the noiseless Glass Identification settings. We find no significant difference between the two optimizers for the noisy KDD Cup and Glass

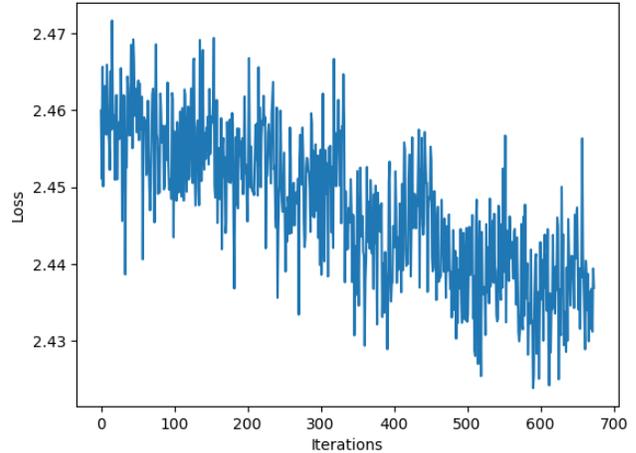


Figure 5: KDD Cup: Nelder-Mead Convergence

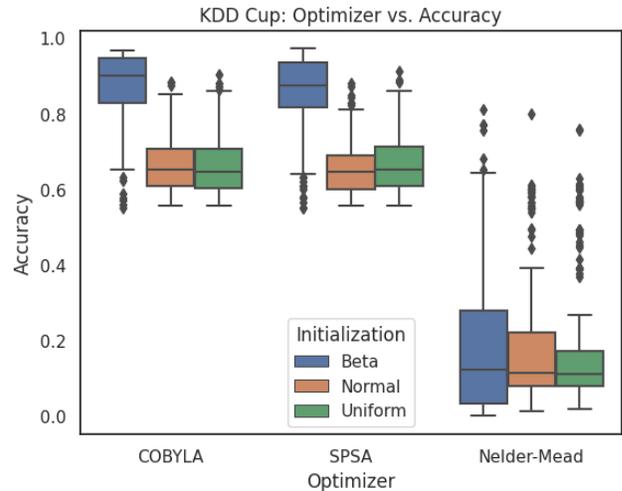


Figure 6: KDD Cup, Noiseless: Optimizer vs. Accuracy

Identification and all Rice settings. Nonetheless, in all cases, they are equally represented among the set of best configurations.

Figure 6 visualizes the performance of the different optimizers for the noiseless experiments on the KDD Cup dataset. Beta initialization leads to significantly better results for SPSA and COBYLA, however not for Nelder-Mead, suggesting that it does not lead to a position in the loss landscape that allows a better optimization path irrespective of the optimizer.

All optimizers are significantly better in the noiseless KDD Cup and Rice settings than in the noisy ones. For the Cover Type dataset, COBYLA and Nelder-Mead perform significantly better in the noisy setting, in contrast to SPSA, which performs better in the noiseless one. We find no significant differences for COBYLA, SPSA to perform better without noise, and Nelder-Mead to perform significantly better with noise for the Glass Identification dataset.

COBYLA is significantly faster than SPSA and Nelder-Mead in all settings, as is SPSA compared to Nelder-Mead.

## 5.4 Ansatz

**Observation:** There are no significant differences between the ansatzes for any dataset, neither in the noiseless nor noisy setting. However, PauliTwoDesign performs slightly worse in mean and is usually less represented among the best configurations.

In particular, while mean and standard deviation are very similar across all ansatzes, PauliTwoDesign has less extreme minima and maxima, exhibiting a more stable performance when compared to the other ansatzes: for the Cover Type noisy setting, the minimum is 5% accuracy, compared to 1%-2% for the other ansatzes, and for the KDD Cup noisy setting, the maximum is 89%, compared to 96%. The same can be observed for the noiseless Glass Identification dataset, with a minimum of 6%, compared to 2% accuracy for the others, and a maximum of 51%, compared to 53%-62%. These patterns are not ever-present, but we observe a tendency in our experiments.

When we directly compare noisy and noiseless settings, we find that all ansatzes are significantly better without noise for the KDD Cup and Rice datasets. We find no differences for EfficientSU2 on Cover Type, PauliTwoDesign to perform better without noise, and RealAmplitudes and TwoLocal to

lead to better results in the noisy setting. For the Glass Identification dataset, we find no differences for EfficientSU2 and TwoLocal, and RealAmplitudes and PauliTwoDesign to perform significantly better without noise.

All ansatzes are significantly faster than EfficientSU2. The longer runtime of EfficientSU2 is explained by the default configuration employing more rotation gates than the other ansatzes, resulting in twice the amount of trainable parameters. Our results imply that these additional parameters do not necessarily lead to better performance. Therefore, it may be favorable to stick to the less parametrized and, hence, faster ansatzes. We visualize the runtimes in Figure 7.

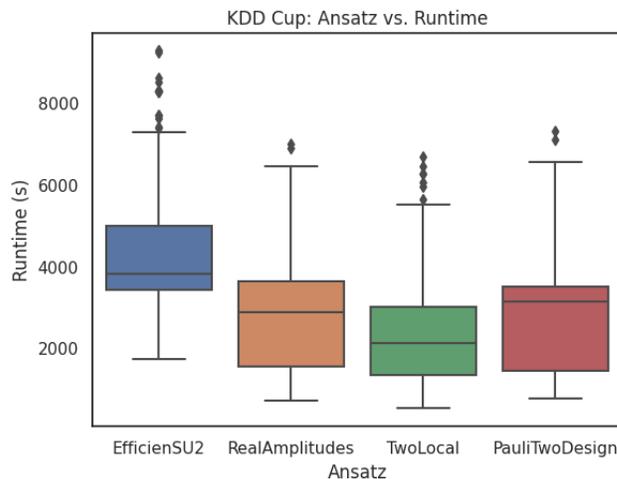


Figure 7: KDD Cup, Noiseless: Ansatz vs. Runtime

## 5.5 Ansatz Entanglement

**Observation:** We find no consistent patterns regarding the performance of the different ansatz entanglement strategies.

For KDD Cup and Cover Type, neither the noiseless nor noisy experiments show any significant differences for the ansatz entanglement strategies. When we compare the noisy and noiseless settings directly, we find no significant differences either. We find some significant differences for the Glass Identification and Rice dataset, however, no consistent patterns. For completeness, we list them in the appendix.

## 5.6 Feature Map

**Observation:** Beta initialization boosts the performance of ZZFeatureMap configurations in most settings significantly. Still, the performance of the feature maps is dependent on the dataset.

ZFeatureMap is significantly better than ZZFeatureMap in the KDD Cup noiseless, and in all Cover Type and Glass Identification settings. Still, we find that the majority of best configurations use ZZFeatureMap in these settings. This sounds counterintuitive at first, however, beta initialization boosts the performance of ZZFeatureMap configurations by 12%-16% accuracy in mean, compared to 2%-5% for ZFeatureMap. Hence, ZZFeatureMap configurations are significantly worse without beta initialization than ZFeatureMap ones, explaining the significance tests. Still, beta initialization boosts the performance of ZZFeatureMap configurations to make up the set of best configurations.

Entanglement in a circuit, as is used in ZZFeatureMap compared to ZFeatureMap, has been linked to BPs [37]. Our results suggest that the proposed beta initialization helps to cope with them, as the performance difference to normal and uniform is significant. We visualize the different feature maps and initializations in Figure 8.

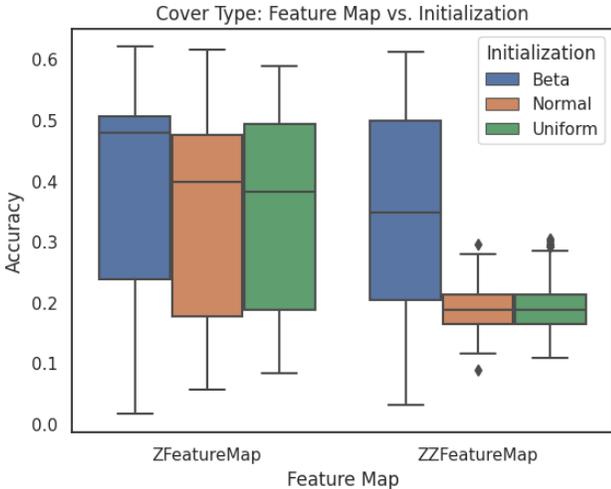


Figure 8: Cover Type, Noiseless: Feature Map vs. Initialization

In contrast, there are no significant differences between

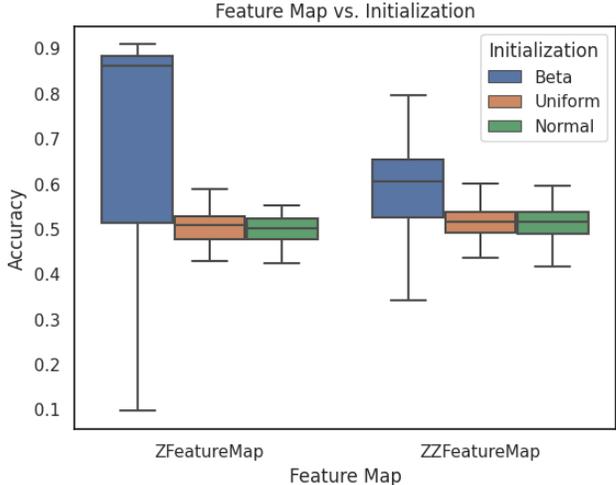


Figure 9: Rice, Noisy: Feature Map vs. Initialization

the feature maps in the KDD Cup noisy setting and all Rice settings. Here, the very best configurations are ZFeatureMap ones. For KDD, we find that beta initialization does not boost the performance of ZZFeatureMap as much. Interestingly, for the Rice dataset, beta initialization boosts ZFeatureMap configurations a lot more (19%-22% accuracy in mean) than ZZFeatureMap ones (7%-9%), i.e., the behavior is reversed (see Figure 9).

When comparing noisy and noiseless settings, we find no consistent patterns. Considering runtime, ZZFeatureMap configurations take significantly longer than ZFeatureMap ones, since the non-linearity and entanglement between the qubits strongly increase the computational complexity of feature encoding.

## 5.7 Feature Map Entanglement

**Observation:** Similarly to the ansatz entanglement, there are no consistent significant differences between the strategies.

We again find significant differences in some experiments, however, no consistent patterns. Still, full entanglement is usually underrepresented among the best configurations. We again refer to the appendix for further information.

We find no consistent significant differences when directly comparing noisy and noiseless settings.

## 5.8 Preprocessing

**Observation:** LDA works significantly better than PCA in most settings or at least leads to a majority of best configurations. This suggests that it may transform the loss landscape in a way that aids optimization.

While PCA outperforms LDA in the KDD noiseless and noisy settings, about 60% of the best configurations use LDA as a preprocessing step in both settings. LDA significantly outperforms PCA for the Cover Type dataset, with 98% of the best configurations using LDA in both the noisy and noiseless setting. Furthermore, we find LDA to perform significantly better in the Glass Identification noiseless (1% accuracy in mean) and noisy (4%) setting. We observe a tendency that the choice for this hyperparameter depends on the dataset, which we elaborate on in the appendix.

Furthermore, in Figure 10, we observe that beta initialization boosts the performance for LDA more than for PCA. The pattern cannot be observed for the Glass Identification dataset, where beta initialization boosts the performance of both preprocessing techniques by 10%-11% in mean for the noiseless and by 5%-6% in the noisy setting.

LDA arranges the features in a way that better separates the different classes, whereas PCA focuses on keeping the variance in the data. Beta initialization boosts the mean performance of PCA configurations for the KDD Cup and Cover Type datasets about 10% in our experiments, compared to 20% for LDA. Our results therefore suggest that LDA can transform the loss landscape such that, given a good starting point, the model is likely to end up in a better solution than with PCA.

When we directly compare the configurations with and without noise, we find that PCA and LDA are significantly better without noise for the KDD Cup dataset (5% difference in accuracy in mean). For the Cover Type dataset, we find no significant differences for LDA, but PCA works significantly better with noise (1% in mean). For Glass Identification, we find no significant differences for PCA, but LDA works significantly better with noise (2%).

When considering the runtime, we find no significant differences for the KDD Cup dataset, but PCA configurations run significantly longer than LDA ones for the Cover Type and Glass Identification datasets. We find that COBYLA PCA configurations take more iterations in mean than LDA ones, as do Nelder-Mead ones, explaining the runtime dif-

ference. Furthermore, the significantly longer convergence could also indicate that the loss landscape of PCA configurations can be more difficult to navigate than LDA ones.

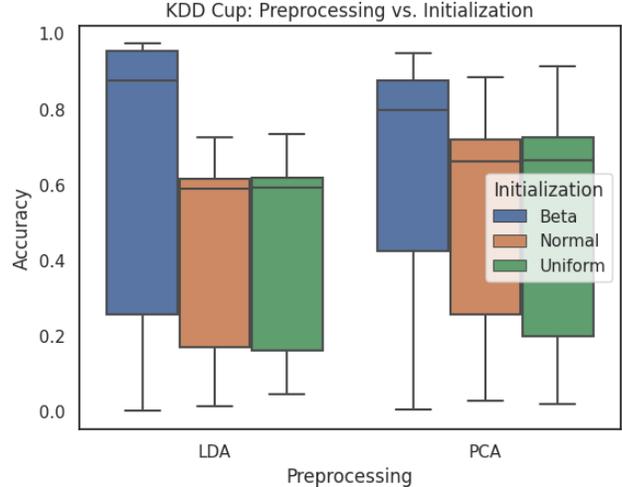


Figure 10: KDD Cup, Noiseless: Preprocessing vs. Initialization

## 5.9 Further Analysis

One of the most prevalent patterns we discover is that the optimizer is the most crucial hyperparameter to set. Using Nelder-Mead in our experiments consistently led to underperforming configurations, independently of any other hyperparameters.

Furthermore, our results show that the initialization strategy plays a crucial role. Nonetheless, we find that the initialization strategy does not necessarily help all optimizers. Figure 11 shows that even when combining initialization and optimizer, the optimizer is the most influential parameter for a model’s performance.

## 6 Discussion

Our experiments gave us valuable insights into tuning hyperparameters for QNNs. The empirical evidence we gathered suggests that the choice of optimizer and initialization method is crucial. In particular, COBYLA and SPSA seem to both be reasonable choices for the optimizer. Since

	Beta				Normal				Static			
	K	C	G	R	K	C	G	R	K	C	G	R
Beta					0.35	7e-4	0.84	0.54	0.45	0.55	0.81	0.22
Normal	0.35	7e-4	0.84	0.54					0.69	7e-5	0.81	0.12
Static	0.45	0.55	0.81	0.22	0.69	7e-5	0.81	0.12				

Table 6: Initialization with Beta Distribution Mean and SD: Significance Tests

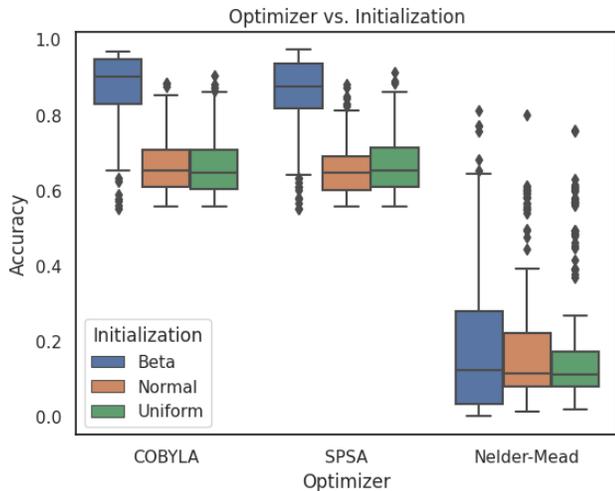


Figure 11: KDD Cup, Noisy: Optimizer vs. Initialization

COBYLA configurations are usually faster, we suggest using it as a starting point. Furthermore, it is favorable to use beta initialization compared to random and normal one, in particular when employing an entangled feature map.

Although ZFeatureMap was often significantly better than ZZFeatureMap, in most settings, the best configurations use ZZFeatureMap. Based on the significant performance differences between beta initialized and non-beta initialized configurations for ZZFeatureMap compared to ZFeatureMap, we hypothesize that entanglement-induced barren plateaus occur and that a clever initialization strategy can indeed help in finding better optimization trajectories.

**Main Finding 3:** While beta distribution initialization works significantly better than all others in our main experiments, post-hoc tests reveal that it is not the probability distribution, but rather the range of values used, that is indicative of the likelihood of encountering a BP. This stresses the importance of exploring the theoretical aspects of parameter initialization further.

After looking at the significant results, we were curious about why exactly beta distribution initialization works so much better than the other two. Therefore, we ran two additional post-hoc tests, initializing all parameters to the mean of the beta distribution, and using a normal distribution with the mean and standard deviation of the chosen beta distribution. We hardly found any significant differences between the three techniques, which we visualize in Table 6. Red/green means the row is significantly worse/better than the column on the specific dataset (K: KDD Cup, C: Cover Type, G: Glass Identification, R: Rice), whereas yellow denotes no significant differences. We test using a Wilcoxon test with a significance level of 0.05 and the numbers denote the p-value.

Furthermore, to our knowledge, there are no indicators of a causal relationship between the distribution of the data and the parameters determining the rotations applied in the circuit (as is assumed in [27]). Therefore, it is an open question whether certain value ranges just lead to better initial positions, or there indeed is a connection between the two, which would be important to study from a theoretical point of view.

The ansatzes perform similarly on average, however, some are more likely to lead to higher-performing outliers. Therefore, we advise starting with RealAmplitudes, which can be used out-of-the-box and takes less time than EfficientSU2. Also, TwoLocal is a good option, however, the entanglement and rotation gates have to be set accordingly. We found hardly any significant differences between entanglement strategies, therefore believe using the default option is

a good start.

We find that the entanglement strategy has a negligible effect on the performance of the model. We hypothesize that entangling the qubits is indeed important to exploit quantum effects, however, the actual way this is done does not seem to have an impact on the performance of the models.

For the KDD Cup and Cover Type dataset, using beta initialization in combination with LDA significantly boosts the performance of the models, therefore, all the best models use LDA. We hypothesize that, as LDA separates the classes during preprocessing already, this, together with advantageous initial starting points from beta initialization, can lead to better trainability of the model.

## 6.1 Threats to Validity

*Scope of the experiment:* While the experiments on the four datasets allowed us to draw valuable conclusions, the results could vary with different ones.

*Reduced dataset:* It is a well-known fact that ML models profit from more data. As we had to limit the training samples, the results could vary when rerunning the experiments with the whole dataset.

*Scalability:* NISQ hardware and the complexity of simulating quantum computers limit today’s experiments. It remains an open research question what happens when the experiments are scaled from tens to hundreds or thousands of qubits.

*Architecture:* We chose the general-purpose QASM simulator [8] for our experiments, however, there are few studies comparing the performance of different simulators. Therefore, the results may differ when using a different one.

## 7 Conclusion and Future Work

In this work, we have emphasized the potential of QML in addressing the challenges of Post-Moore era. However, the complexity of the models, combined with few hyperparameter tuning studies with only limited scope to this date, hinder the adoption and further exploration. Therefore, we have collected data about the performance of QML models on various datasets using different hyperparameter configurations with and without noise and evaluated our results rigorously. Our results support researchers and practitioners who want to explore QML and potential applications further, by providing starting points for tuning the models.

Besides collecting data, our experiments on real-world datasets provided us with evidence that initializing using the Gaussian distribution, as proposed in [56], does not lead to advantages over initializing using a uniform distribution in training the models. Furthermore, in contrast to the conjecture from [27], we found in additional experiments that the beta distribution itself does not necessarily lead to better initial points and therefore to better predictive performance of the models. Indeed, we mostly found no significant differences when initializing using a normal distribution with the same mean and standard deviation as the beta distribution, and initializing all parameters to the mean of the beta distribution.

Nonetheless, given the importance of theoretically studying these models further, in particular with respect to trainability issues such as BPs, we plan to further explore the model space and loss landscapes. While it seems that clever initialization can mitigate BPs to some extent, we want to consider why certain initial parameter ranges result in better models overall. Also, we plan to extend this study by considering different types of datasets and different types of quantum architectures.

## Acknowledgements

This work has been partially funded through the Rucon project (Runtime Control in Multi Clouds), Austrian Science Fund (FWF): Y904-N31 START-Programm 2015, by the CHIST-ERA grant CHIST-ERA-19-CES-005, Austrian Science Fund (FWF), Standalone Project Transprecise Edge Computing (Triton), Austrian Science Fund (FWF): P 36870-N, and by Flagship Project HPQC (High Performance Integrated Quantum Computing) # 897481 Austrian Research Promotion Agency (FFG). We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

## References

- [1] Cezar-Mihail Alexandru, Ella Bridgett-Tomkinson, Noah Linden, Joseph MacManus, Ashley Montanaro, and Hannah Morris. Quantum speedups of some general-purpose numerical optimisation algorithms.

- Quantum Science and Technology*, 5(4):045014, sep 2020.
- [2] Raúl Berganza Gómez, Corey O’Meara, Giorgio Cortiana, Christian B. Mendl, and Juan Bernabé-Moreno. Towards autoqml: A cloud-based automated circuit architecture search framework. In *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, pages 129–136, Honolulu, HI, USA, 2022. IEEE.
- [3] Jock Blackard. Coverttype. UCI Machine Learning Repository, 1998. DOI: <https://doi.org/10.24432/C50K5N>.
- [4] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, August 2021.
- [5] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12(1):1791, Mar 2021.
- [6] Ilkay Cinar and Murat Koklu. Classification of rice varieties using artificial intelligence methods. *International Journal of Intelligent Systems and Applications in Engineering*, 7:188–194, 09 2019.
- [7] M. Crippa, D. Guizzardi, M. Banja, E. Solazzo, M. Muntean, E. Schaaf, F. Pagani, F. Monforti-Ferrario, J. Olivier, R. Quadrelli, A. Risquezz Martin, P. Taghavi-Moharamli, G. Grassi, S. Rossi, D. Jacome Felix Oom, A. Branco, J. San-Miguel-Ayanz, and E. Vignati. Co2 emissions of all world countries - 2022 report, 2022.
- [8] Andrew W. Cross, Lev S. Bishop, John A. Smolin, and Jay M. Gambetta. Open quantum assembly language, 2017.
- [9] Gennaro De Luca. A Survey of NISQ Era Hybrid Quantum-Classical Machine Learning Research. *Journal of Artificial Intelligence and Technology*, 2(1):9–15, 2022.
- [10] Amir H. Gandomi, Fang Chen, and Laith Abualigah. Machine Learning Technologies for Big Data Analytics. *Electronics*, 11(3):421, January 2022.
- [11] B. German. Glass Identification. UCI Machine Learning Repository, 1987. DOI: <https://doi.org/10.24432/C5WW2P>.
- [12] Takahiro Goto, Quoc Hoan Tran, and Kohei Nakajima. Universal Approximation Property of Quantum Machine Learning Models in Quantum-Enhanced Feature Spaces. *Physical Review Letters*, 127(9):090506, August 2021.
- [13] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, December 2019.
- [14] Laszlo Gyongyosi and Sandor Imre. A Survey on quantum computing technology. *Computer Science Review*, 31:51–71, February 2019.
- [15] Stuart Hadfield, Zihui Wang, Bryan O’Gorman, Eleanor G. Rieffel, Davide Venturelli, and Rupak Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2), 2019.
- [16] Juan Kenyhy Hancoco-Quispe, Jordan Piero Borda-Colque, and Fred Torres-Cruz. Quantum machine learning applied to the classification of diabetes. *ArXiv*, abs/2301.00109, 2022.
- [17] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, March 2019.
- [18] Zoë Holmes, Kunal Sharma, M. Cerezo, and Patrick J. Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, 3:010313, Jan 2022.
- [19] He-Liang Huang, Xiao-Yue Xu, Chu Guo, Guojing Tian, Shi-Jie Wei, Xiaoming Sun, Wan-Su Bao, and Gui-Lu Long. Near-term quantum computing techniques: Variational quantum algorithms, error mitigation, circuit compilation, benchmarking and classical

- simulation. *Science China Physics, Mechanics & Astronomy*, 66(5):250302, April 2023.
- [20] Nisheeth Joshi, Pragya Katyayan, and Syed Afroz Ahmed. Comparing Classical ML Models with Quantum ML Models with Parametrized Circuits for Sentiment Analysis Task. *Journal of Physics: Conference Series*, 1854(1):012032, April 2021.
- [21] Nisheeth Joshi, Pragya Katyayan, and Syed Afroz Ahmed. Evaluating the Performance of Some Local Optimizers for Variational Quantum Classifiers. *Journal of Physics: Conference Series*, 1817(1):012015, March 2021.
- [22] Pragya Katyayan and Nisheeth Joshi. Supervised Question Classification on SelQA Dataset Using Variational Quantum Classifiers. In *International Conference on Innovative Computing and Communications*, volume 492, pages 695–706. Springer Nature Singapore, Singapore, 2023.
- [23] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The uci machine learning repository, 2023.
- [24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- [25] Toshiaki Koike-Akino, Pu Wang, and Ye Wang. Autotqml: Automated quantum machine learning for wi-fi integrated sensing and communications. In *2022 IEEE 12th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pages 360–364, Trondheim, Norway, 2022. IEEE.
- [26] M. Koklu, I. Cinar, and Y. S. Taspinar. Rice (Cammeo and Osmancik). UCI Machine Learning Repository, 2019.
- [27] Ankit Kulshrestha and Ilya Safro. Beinit: Avoiding barren plateaus in variational quantum algorithms. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 197–203, Broomfield, CO, USA, 2022. IEEE.
- [28] Jeffrey C. Lagarias, James A. Reeds, Margaret H. Wright, and Paul E. Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147, 1998.
- [29] Xinwei Lee, Yoshiyuki Saito, Dongsheng Cai, and Nobuyoshi Asai. Parameters fixing strategy for quantum approximate optimization algorithm. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 10–16, 2021.
- [30] Richard Lippmann, Joshua W Haines, David J Fried, Jonathan Korba, and Kumar Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, 34(4):579–595, October 2000.
- [31] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning, 2013.
- [32] Javier Mancilla and Christophe Pere. A Preprocessing Perspective for Quantum Machine Learning Classification Advantage in Finance Using NISQ Algorithms. *Entropy*, 24(11):1656, November 2022.
- [33] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1):4812, Nov 2018.
- [34] Nimish Mishra, Manik Kapil, Hemant Rakesh, Amit Anand, Nilima Mishra, Aakash Warke, Soumya Sarkar, Sanchayan Dutta, Sabhyata Gupta, Aditya Prasad Dash, Rakshit Gharat, Yagnik Chatterjee, Shivarati Roy, Shivam Raj, Valay Kumar Jain, Shree-ram Bagaria, Smit Chaudhary, Vishwanath Singh, Rituparna Maji, Priyanka Dalei, Bikash K. Behera, Sabyasachi Mukhopadhyay, and Prasanta K. Panigrahi. Quantum Machine Learning: A Review and Current Status. In *Data Management, Analytics and Innovation*, volume 1175, pages 101–145. Springer Singapore, Singapore, 2021.
- [35] Charles Moussa, Henri Calandra, and Vedran Dunjko. To quantum or not to quantum: towards algorithm selection in near-term quantum optimization. *Quantum Science and Technology*, 5(4):044009, oct 2020.
- [36] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, January 1965.
- [37] Carlos Ortiz Marrero, Mária Kieferová, and Nathan Wiebe. Entanglement-induced barren plateaus. *PRX Quantum*, 2:040316, Oct 2021.

- [38] Davide Pastorello. *Concise guide to quantum machine learning*. Springer, Singapore, 2023. OCLC: 1362515386.
- [39] Ilya Piatrenka and Marian Rusek. Quantum Variational Multi-class Classifier for the Iris Data Set. In *Computational Science – ICCS 2022*, volume 13353, pages 247–260. Springer International Publishing, Cham, 2022.
- [40] M. J. D. Powell. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. In *Advances in Optimization and Numerical Analysis*, pages 51–67. Springer Netherlands, Dordrecht, 1994.
- [41] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.
- [42] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023.
- [43] Patrick Reberntrost, Masoud Mohseni, and Seth Lloyd. Quantum Support Vector Machine for Big Data Classification. *Physical Review Letters*, 113(13):130503, September 2014.
- [44] Salonik Resch and Ulya R. Karpuzcu. Benchmarking Quantum Computers and the Impact of Quantum Noise. *ACM Computing Surveys*, 54(7):1–35, September 2022.
- [45] Maria Schuld and Nathan Killoran. Is Quantum Advantage the Right Goal for Quantum Machine Learning? *PRX Quantum*, 3(3):030101, July 2022.
- [46] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3):032430, March 2021.
- [47] Martin Schulz, Dieter Kranzlmüller, Laura Brandon Schulz, Carsten Trinitis, and Josef Weidendorfer. On the inevitability of integrated hpc systems and how they will change hpc system operations. In *Proceedings of the 11th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*, HEART ’21, New York, NY, USA, 2021. Association for Computing Machinery.
- [48] Daniel Sierra-Sosa, Soham Pal, and Michael Telahun. Data rotation and its influence on quantum encoding. *Quantum Information Processing*, 22(1):89, January 2023.
- [49] J. Spall. An Overview of the Simultaneous Perturbation Method for Efficient Optimization. *Johns Hopkins Apl Technical Digest*, 19(4):482–492, 1998.
- [50] Hatma Suryotrisongko and Yasuo Musashi. Hybrid Quantum Deep Learning and Variational Quantum Classifier-Based Model for Botnet DGA Attack Detection. *International Journal of Intelligent Engineering and Systems*, 15(3):215–224, June 2022.
- [51] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shrutu Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kamradur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.
- [52] Guillaume Verdon, Michael Broughton, Jarrod R. McClean, Kevin J. Sung, Ryan Babbush, Zhang Jiang, Hartmut Neven, and Masoud Mohseni. Learning to learn with quantum neural networks via classical neural networks, 2019.
- [53] Samson Wang, Enrico Fontana, M. Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio, and Patrick J. Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature Communications*, 12(1):6961, Nov 2021.

- [54] Roeland Wiersema, Cunlu Zhou, Yvette de Sereville, Juan Felipe Carrasquilla, Yong Baek Kim, and Henry Yuen. Exploring entanglement and optimization within the hamiltonian variational ansatz. *PRX Quantum*, 1:020319, Dec 2020.
- [55] Madita Willsch, Dennis Willsch, Fengping Jin, Hans De Raedt, and Kristel Michielsen. Benchmarking the quantum approximate optimization algorithm. *Quantum Information Processing*, 19(7):197, Jun 2020.
- [56] Kaining Zhang, Liu Liu, Min-Hsiu Hsieh, and Dacheng Tao. Escaping from the barren plateau via gaussian initializations in deep variational quantum circuits. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 18612–18627, New Orleans, LA, USA, 2022. Curran Associates, Inc.
- [57] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Phys. Rev. X*, 10:021067, Jun 2020.

## Appendix

In the following, we will elaborate further on the results of the significance tests we conducted. Red/green means that the row is significantly worse/better than the column. yellow denotes no significant differences and the values are the p-values, rounded to the fourth decimal.

### Initialization

Table 7 shows the comparison of the three main initialization strategies we tested. We want to highlight that beta initialization was significantly better in *all* experiments conducted than all other strategies. Furthermore, we find significant differences between normal and uniform initialization in only one setting.

### Preprocessing

Table 8 compares the two preprocessing techniques. In particular, we would like to highlight that classical preprocess-

ing, in our experiments, has a significant impact on the results. We find significant differences in *all* experiments, and the results always hold for the noisy and noiseless experiments. This suggests that the dataset is an important factor.

### Feature Map Entanglement

Table 9 compares the feature map entanglement. We do not find patterns that are ever-present, however, full entanglement is never significantly better than any other entanglement strategy and is often outperformed. This stands in contrast to linear and pairwise entanglement, which are both never outperformed by any other strategy.

### Ansatz Entanglement

Table 10 compares the ansatz entanglement strategies. Interestingly, we see the tendency of full entanglement being worse than others from the feature map entanglement reversed, i.e., it is either significantly better or not different from others. Furthermore, sca entanglement never provided any advantages in our experiments.

		Beta				Normal				Uniform			
		K	C	G	R	K	C	G	R	K	C	G	R
Beta	P					0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	N					0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Normal	P	0.0000	0.0000	0.0000	0.0000					0.3375	0.2777	0.3385	0.4354
	N	0.0000	0.0000	0.0000	0.0000					0.0000	0.5102	0.6722	0.4412
Uniform	P	0.0000	0.0000	0.0000	0.0000	0.3375	0.2777	0.3385	0.4354				
	N	0.0000	0.0000	0.0000	0.0000	0.0000	0.5102	0.6722	0.4412				

Table 7: Initialization: Significance Tests

		LDA			PCA		
		K	C	G	K	C	G
LDA	P				0.0000	0.0000	0.0008
	N				0.0000	0.0000	0.0000
PCA	P	0.0000	0.0000	0.0008			
	N	0.0000	0.0000	0.0000			

Table 8: Preprocessing: Significance Tests

		circular				full				linear			
		K	C	G	R	K	C	G	R	K	C	G	R
circular	P					0.0001	0.018	0.2119	0.7301	0.1493	0.0000	0.4048	0.5988
	N					0.0000	0.5443	0.2441	0.1297	0.1537	0.0000	0.4214	0.8146
full	P	0.0001	0.018	0.2119	0.7301					0.0000	0.0000	0.0033	0.5714
	N	0.0000	0.5443	0.2441	0.1297					0.0000	0.0000	0.6002	0.1949
linear	P	0.1493	0.0000	0.4048	0.5988	0.0000	0.0000	0.0033	0.5714				
	N	0.1537	0.0000	0.4214	0.8146	0.0000	0.0000	0.6002	0.1949				
pairwise	P	0.0503	0.0000	0.9165	0.1658	0.0000	0.0000	0.1663	0.4471	0.7099	0.2139	0.4942	0.456
	N	0.5574	0.0000	0.4476	0.7883	0.0000	0.0000	0.8776	0.229	0.8585	0.792	0.7253	0.3521
sca	P	0.7064	0.3664	0.422	0.6358	0.0001	0.0012	0.3264	0.8701	0.2298	0.0000	0.1139	0.5581
	N	0.3778	0.4753	0.3436	0.8693	0.0000	0.1038	0.4568	0.0642	0.023	0.0001	0.8478	0.8006
		pairwise				sca							
circular	P	0.0503	0.0000	0.9165	0.1658	0.7064	0.3664	0.422	0.6358				
	N	0.5574	0.0000	0.4476	0.7883	0.3778	0.4753	0.3436	0.8693				
full	P	0.0000	0.0000	0.1663	0.4471	0.0001	0.0012	0.3264	0.8701				
	N	0.0000	0.0000	0.8776	0.229	0.0000	0.1038	0.4568	0.0642				
linear	P	0.7099	0.2139	0.4942	0.456	0.2298	0.0000	0.1139	0.5581				
	N	0.8585	0.792	0.7253	0.3521	0.023	0.0001	0.8478	0.8006				
pairwise	P					0.0093	0.0000	0.3395	0.3111				
	N					0.2028	0.0001	0.7977	0.8099				
sca	P	0.0093	0.0000	0.3395	0.3111								
	N	0.2028	0.0001	0.7977	0.8099								

Table 9: Feature Map Entanglement: Significance Tests

		circular				full				linear			
		K	C	G	R	K	C	G	R	K	C	G	R
circular	P					0.7625	0.1441	0.0594	0.0042	0.9804	0.7872	0.0623	0.2437
	N					0.9271	0.2133	0.6949	0.0000	0.4918	0.3285	0.5024	0.0085
full	P	0.7625	0.1441	0.0594	0.0042					0.8075	0.0922	0.8753	0.1513
	N	0.9271	0.2133	0.6949	0.0000					0.4153	0.0143	0.291	0.0243
linear	P	0.9804	0.7872	0.0623	0.2437	0.8075	0.0922	0.8753	0.1513				
	N	0.4918	0.3285	0.5024	0.0085	0.4153	0.0143	0.291	0.0243				
pairwise	P	0.9681	0.1016	0.5506	0.3223	0.7946	0.5155	0.3938	0.2417	0.9266	0.0746	0.4108	0.9789
	N	0.7013	0.0964	0.8638	0.0571	0.6871	0.3874	0.8669	0.3273	0.842	0.0131	0.5402	0.6781
sca	P	0.3545	0.4493	0.2169	0.6482	0.2225	0.4811	0.0006	0.0011	0.3923	0.3062	0.001	0.101
	N	0.3656	0.5908	0.0017	0.1153	0.315	0.0654	0.0003	0.0022	0.8758	0.6311	0.0073	0.3553
		pairwise				sca							
circular	P	0.9681	0.1016	0.5506	0.3223	0.3545	0.4493	0.2169	0.6482				
	N	0.7013	0.0964	0.8638	0.0571	0.3656	0.5908	0.0017	0.1153				
full	P	0.7946	0.5155	0.3938	0.2417	0.2225	0.4811	0.0006	0.0011				
	N	0.6871	0.3874	0.8669	0.3273	0.315	0.0654	0.0003	0.0022				
linear	P	0.9266	0.0746	0.4108	0.9789	0.3923	0.3062	0.001	0.101				
	N	0.842	0.0131	0.5402	0.6781	0.8758	0.6311	0.0073	0.3553				
pairwise	P					0.5738	0.259	0.1396	0.1724				
	N					0.7987	0.0226	0.0137	0.356				
sca	P	0.5738	0.259	0.1396	0.1724								
	N	0.7987	0.0226	0.0137	0.356								

Table 10: Ansatz Entanglement: Significance Tests